Journal of Machine Learning Research ? (20??) ?-? Submitted 2/10; Published ?/0?

Sparse PCA by Iterative Elimination Algorithm

Yang Wang YWANG@MATH.MSU.EDU
Qiang Wu WUQIANG@MATH.MSU.EDU
Department of Mathematics, Michigan State University, East Lansing, MI 48824, USA.

Editor:

Abstract

In this paper we proposed an iterative elimination algorithm for sparse principal component analysis. It recursively eliminates variables according to certain criterion that aims to minimize the loss of explained variance, and reconsiders the sparse principal component analysis problem until the desired sparsity is achieved. Two criteria, the approximated minimal variance loss (AMVL) criterion and the minimal absolute value criterion, are proposed to select the variables eliminated in each iteration. Deflation techniques are discussed for multiple principal components computation. The effectiveness is illustrated by both simulations on synthetic data and applications on real data.

Keywords: Sparse Principal Component Analysis, Iterative Elimination, Approximated Minimal Variance Loss Criterion, Deflation.

1. Introduction

Principal component analysis (PCA) is a popular dimension reduction technique. It is widely used for data compression, statistical modeling, and data visualization in sciences and engineering (Jolliffe, 2002). The task of PCA is to find principal components (PCs) which are the linear combinations of the variables and explain maximum variance of the data. The main advantages of PCA lie in the minimal loss of information, uncorrelated structures, and easy interpretability of linear combinations.

A well known drawback of PCA is the lack of sparsity. Usually all loadings of the principal components are nonzero. From data analysis perspective, sparsity is desirable for reduced computational time and better generalization performance. From modeling perspective, although the interpretability of linear combinations is usually easy for low dimensional data, it could become much more difficult when the number of variables becomes huge, for example, in the gene expression data analysis. In order to overcome this difficulty and introduce sparsity, numerous methods have been developed (Cadima and Jolliffe, 1995; Jolliffe et al., 2003; Zou et al., 2006; Moghaddam et al., 2006; Sriperumbudur et al., 2007; Zass and Shashua, 2007; d'Aspremont et al., 2008; Shen and Huang, 2008; Journée et al., 2008). An ad hoc method that thresholds loadings with small absolute values to 0, termed as "simple thresholding", is introduced in (Cadima and Jolliffe, 1995). Advanced methods include the SCoLTLASS (Jolliffe et al., 2003), lasso based sparse PCA (SPCA, Zou et al. 2006), direct sparse PCA (DSPCA, d'Aspremont et al. 2007), Greedy sparse PCA (GSPCA, Moghaddam et al. 2006), generalized power method (GPower, Journée et al. 2008), sparse PCA by d.c. programming (DC-SPCA, Sriperumbudur et al. 2007, 2009), etc.

In this paper we introduce a new approach for sparse PCA which we call the *Iterative Elimination (IE)* algorithm. This approach is motivated by the well known Recursive Feature Elimination (RFE) technique in learning theory and the the simple thresholding method. In the iterative elimination algorithm variables are recursively eliminated through a ranking criterion, which can be either the minimal absolute value criterion or the more sophisticated *approximated minimal variance loss (AMVL)* criterion that will be introduced later in the paper. A main criticism on simple thresholding method has been that it can easily be misleading in many cases, where smaller loadings in the PCA do not always mean less significance in explaining variance. Many examples have been used to demonstrate such deficiency and the resulting sub-optimality of simple thresholding. However, it is also true that in general the variable with the smallest loading has little possibility to be among the variables that have the greatest influence. This motivates the idea to remove the variable with smallest loading and reconsider the sparse PCA in the reduced space. Based on this heuristics the iterative elimination algorithm for sparse PCA eliminates one or a small portion of the variables at one time and repeats this procedure until the desired sparsity is achieved. We will demonstrate that this approach is simple and efficient, and yet it is powerful and compares well with other state-of-the-art approaches.

2. Iterative elimination

In this section we introduce and discuss our iterative elimination approach in detail. We focus on the computation of the first sparse PC since the latter ones can be obtained via deflation techniques.

Denote by $\Sigma = (\Sigma_{ij})$ a p dimensional covariance matrix. It has to be symmetric and positive definite. For a vector $\mathbf{v} = (v_1, \ldots, v_p)^\top \in \mathbb{R}^p$, $\|\mathbf{v}\|$ is the Euclidian norm and $\|\mathbf{v}\|_0$ denotes the number of nonzero elements in $\mathbf{v}$.

Recall in the ordinary PCA, the first PC is the unit vector along which maximum variance is explained:

$$\mathbf{v}_1 = \arg \max_{\|\mathbf{v}\|=1} \mathbf{v}^\top \Sigma \mathbf{v}. \tag{1}$$

It turns out $\mathbf{v}_1$ is the eigenvector associated with the largest eigenvalue $\lambda_1(\Sigma)$ of Σ. All the loadings $v_{1,i}$, $i = 1, \ldots, p$, of $\mathbf{v}_1$ are usually nonzero. The sparse PCA seeks the 'pseudo eigenvector' with desired sparsity:

$$\mathbf{u}_1 = \arg \max_{\|\mathbf{u}\|=1} \mathbf{u}^\top \Sigma \mathbf{u} \quad \text{subject to } \|\mathbf{u}\|_0 \le k \tag{2}$$

where $k < p$ is a positive integer. This is an NP hard problem. Most of state-of-the-art sparse PCA methods in the literature try to relax the sparsity constraint and solve approximated optimization problems.

Our iterative elimination approach follows from a different spirit. It relies on an elimination criterion and adopt a recursive variable elimination procedure.

In the subsequent sections of this paper, we will use the following notations: for a symmetric matrix A, $\mathbf{v}_\ell(A)$ and $\lambda_\ell(A)$ represent the ℓ-th largest eigenvectors and eigenvalues; $\mathbf{u}_\ell(A)$ and $\sigma_\ell(A)$ represent the sparse pseudo eigenvectors and the corresponding explained variances, respectively. When $A = \Sigma$ is the covariance matrix under consideration and whenever there is no confusion we use the notations $\mathbf{v}_\ell$, λ_ℓ, $\mathbf{u}_\ell$, and σ_ℓ for simplicity.

2.1 Thresholding criteria

In simple thresholding, $\mathbf{u}_1$ is obtained directly from $\mathbf{v}_1$ by thresholding the loadings with smallest absolute values to zero and normalization. The underlying premise of this method is that the variables with smaller loadings contribute less to the variance explained by $\mathbf{v}_1$. However, many examples have demonstrated that this is not often false, but is highly unreliable as a technique for sparse PCA.

Strictly speaking, there is no exact monotone relation between the contribution of the variables and the corresponding absolute values of loadings. But they are indeed closely related. Denote by $\Sigma^{\setminus i}$ the $p-1$ dimensional matrix given by Σ with the i-th row and column deleted. Then the following conclusion is true.

Proposition 1 *For all $i = 1, \ldots, p$,*

$$\lambda_1(\Sigma) - \lambda_1(\Sigma^{\setminus i}) \leq \frac{v_{1,i}^2(\lambda_1(\Sigma) - \Sigma_{ii})}{1 - v_{1,i}^2}.$$

This proposition provides an upper bound for the variance loss if a variable is removed. The bound depends not only on the loading $v_{1,i}$ but also the difference $\lambda_1 - \Sigma_{ii}$. Therefore, only if $\lambda_1 \gg \Sigma_{ii}$ for all i, the absolute values of loadings might be good measures for the importance of the variables. Otherwise, it may be misleading.

Proposition 1 leads to a new elimination criterion. Variables can be ranked according to the bounds on variance loss. We refer it to the *approximated minimal variance loss (AMVL)* criterion. Note this criterion is not exact either. However, we will show in Section 3 that it usually yields better results than the conventional thresholding which will be termed as *minimal absolute value (MAV)* criterion.

2.2 Iterative elimination algorithm

The idea of iterative elimination is motivated by the recursive feature elimination (RFE) technique. RFE is a backward feature selection method and has been introduced to the support vector machines in Guyon et al. (2002). The basic idea of this technique is as follows: The direct variable ranking (according to some estimated criterion) may be rough. But the variable ranked as least important is seldom among the top important ones. So we can remove it first and re-rank the remaining variables. Due to the reduction of the dimension, the ranking may improve.

In sparse PCA, it is time consuming to compute the exact variance loss for all variables. We have to adopt the estimated quantities as the ranking criterion, either the absolute values of loadings or the upper bounds of variance losses. Under these criteria, if a variable ranked as the least important one, it does not necessarily contribute the least variance. But we do know that its contribution is relatively small and removing it will not result in large variance loss. So it is probably not among the desired top k important variables. This is the setting where RFE can works well.

Next we describe our iterative elimination algorithm for sparse PCA as follows:

1. Initialize $\Sigma^{(t)} = \Sigma$, $S_t = \{1, \dots, p\},\ R_t = \emptyset$.
2. In step t,
   - compute the largest eigenvalue $\lambda_1^{(t)}$ and corresponding eigenvector $\mathbf{v}_1^{(t)}$;
   - find the least important variable i_t: if MAV criterion is used,
   $$i_t = \arg\min_{i \in S_t} |v_{1,i}^{(t)}|$$
   or, if AMVL criterion is used
   $$i_t = \arg\min_{i \in S_t} \frac{(v_{1,i}^{(t)})^2 \left(\lambda_1^{(t)} - \Sigma_{ii}^{(t)}\right)}{1 - (v_{1,i}^{(t)})^2};$$
   - update $S_{t+1} = S_t \backslash i_t$, $R_{t+1} = R_t \bigcup \{i_t\}$, and $\Sigma^{(t+1)} = \Sigma(S_{t+1}, S_{t+1})$;
3. Stop until $|S_t| = k$ and output $\mathbf{u}_1$ with $\mathbf{u}_1(S_t) = \mathbf{v}_1^{(t)}$ and $\mathbf{u}_1(R_t) = 0$.

In practice, to speed up the computations, one can eliminate more variables in one step. The number of eliminated variables can be set according to a number of criteria. For example it can be a fixed number or a fixed percentage of the remaining variables, or it can be set dynamically by variance loss. In a problem with very large dimension, such as gene expression data where p is thousands or tens of thousands, the latter method is suggested because elimination of hundreds of variables in one step usually results in little variance loss. When the dimension becomes smaller, the step size should also be small to avoid false selection of variables.

Iterative elimination is viable for problems involving very large dimensions. A "large p, small n" problem refers to a problem with very high dimensional data but limited observations. This setting constantly appears in gene expression data analysis and images processing. For such problems we do not need to compute the covariance matrix. Instead we work on the centered data matrix $\bar{X}$ and in each iteration we only need to compute its largest singular value and the corresponding singular vectors.

2.3 An alternative problem

We argue that the problem (2) finding sparse principal components with given sparsity is not the most useful setting in practice, although it is the mostly studied setting for the sparse PCA research in the literature. Recall in data analysis by PCA, one expects to explain as much variance as possible. Sparse PCA should guarantee enough variability in the data is kept for explanation purposes as it tries to reduce the number of explanatory variables. Obviously a pre-specified sparsity level is not good for this purpose unless it can be easily determined.

We propose an alternative sparse PCA setting. It finds the sparsest principal components with enough variance explained:

$$\mathbf{u}_1 = \arg \min_{\|\mathbf{u}\|=1} \|\mathbf{u}\|_0 \quad \text{subject to } \mathbf{u}^\top \Sigma \mathbf{u} \geq \rho\lambda_1 \tag{3}$$

where $0 < \rho < 1$ controls the variance explained.

A major advantage the iterative elimination algorithm has over several of the other algorithms is that it can be adapted for this alternative problem. All we need to do is to check the variance in each step and stop before the variance drops below the required level. In an extremely large dimensional problem where many variables will need to be removed in one step, we can use the AMVL criterion to guarantee that we do not over eliminate variables.

2.4 Computational considerations

The computational complexity for computing the largest eigenvalues and the corresponding eigenvectors required $O(p^3)$ for a p dimensional matrix. This seems to result in computational complexity $O(p^4)$ to obtain the whole path of sparse eigenvectors, i.e., the eigenvectors with all possible sparsity $k = 1, \ldots, p$. However, notice that in each iteration, the new matrix is obtained by delete one row and one column. The difference is relatively small, especially when the dimension is very large. A fast update is possible by eigen-decomposition algorithms such as power method or Raleigh quotient method (Golub and Loan, 1983). We illustrate this by considering the power method.

Power method is an algorithm to compute the largest eigenvalue and the corresponding eigenvector. Recall that, given a positive semi-definite matrix A and an initial vector q_0, the power method iteratively computes

$$q_{\ell+1} = Aq_\ell / \|Aq_\ell\|.$$

If q_0 is not deficient (i.e. q_0 is not orthogonal to $\mathbf{v}_1(A)$) , then q_ℓ converges to an eigenvector associated to the largest eigenvalue $\mathbf{v}_1(A)$. The convergence depends on how close q_0 is to $\mathbf{v}_1(A)$ and how small is the ratio $\frac{\lambda_2(A)}{\lambda_1(A)}$ (Golub and Loan, 1983). In general the power method is not necessarily very efficient, especially for large matrices. But when q_0 is close to $\mathbf{v}_1(A)$ the convergence is usually fast. As a result, the power method can be very efficient in our setting due to the availability of a very good initial vector. To see this, consider the matrices $\Sigma^{(t)}$ and $\Sigma^{(t+1)}$. Denote the dimension of $\Sigma^{(t)}$ as p_t and, without loss of generality, assume $i_t = p_t$. This means $\Sigma^{(t+1)}$ is obtained by deleting the last row and column of $\Sigma^{(t)}$:

$$\Sigma^{(t)} = \begin{pmatrix} \Sigma^{(t+1)} & * \\ * & \Sigma^{(t)}_{p_t,p_t} \end{pmatrix}.$$

Now consider $\tilde{\Sigma}^{(t+1)}$ defined by

$$\tilde{\Sigma}^{(t+1)} = \begin{pmatrix} \Sigma^{(t+1)} & \mathbf{0} \\ \mathbf{0} & \Sigma^{(t)}_{p_t,p_t} \end{pmatrix}.$$

Note that $\tilde{\Sigma}^{(t+1)}$ is positive semi-definite and its difference from $\Sigma^{(t)}$ is comparatively small, particularly when p_t is large. Thus it is expected that $\mathbf{v}_1^{(t)} = \mathbf{v}_1(\Sigma^{(t)})$ is a good initial guess for computing the eigenvector of $\tilde{\Sigma}^{(t+1)}$ and $\lambda_1(\Sigma^{(t)})$ is close to $\lambda_1(\tilde{\Sigma}^{(t+1)})$. Also, $v^{(t)}_{1,p_t}$ has the smallest absolute values usually means $\Sigma^{(t)}_{p_t,p_t} \ll \lambda_1(\Sigma^{(t)})$. In this case $\Sigma^{(t)}_{p_t,p_t} \ll \lambda_1(\tilde{\Sigma}^{(t+1)})$, which implies $\lambda_1(\Sigma^{(t+1)}) = \lambda_1(\tilde{\Sigma}^{(t+1)})$ and

$$(\mathbf{v}_1^{(t+1)}, 0) = \mathbf{v}_1(\tilde{\Sigma}^{(t+1)}). \tag{4}$$

This equality tells us that the last element of $\mathbf{v}_1(\tilde{\Sigma}^{(t+1)})$ must be zero. Therefore, we expect the the vector $\frac{1}{1-(v^{(t)}_{1,p_t})^2}(v^{(t)}_{1,1}, \ldots, v^{(t)}_{1,p_{t-1}}, 0)$ is also a good initial guess for $\mathbf{v}_1(\tilde{\Sigma}^{(t+1)})$. By (4) again,

$$q_0^{(t+1)} = \frac{1}{1-(v^{(t)}_{1,p_t})^2}(v^{(t)}_{1,1}, \ldots, v^{(t)}_{1,p_{t-1}})$$

serves as a good initial vector for the computation of the first eigenvector of $\Sigma^{(t+1)}$ via the power method.

Various simulations show that just a few iterations typically achieve very high accuracy. Suppose that it takes $\ell^{(t+1)}$ iterations to compute $\mathbf{v}_1^{(t+1)}$ from $q_0^{(t+1)}$ using the power method, the computational complexity is then $O(\ell^{(t+1)}(p_t-1)^2)$ to obtain $\mathbf{v}_1^{(t+1)}$. Let ℓ^* denote the maximum of all $\ell^{(t)}$. Then the total computational complexity to obtain the whole path of the first sparse principal component will be $O(\ell^* p^3)$.

Iterative elimination is viable for a “large p, small n” problem. In each step we can use the power method to find the leading singular vector with a computational complexity $O(\ell^{(t)}p_t n)$ for data matrix of dimension p_t. This will result in total complexity of $O(\ell^* p^2 n)$ for computing the sparse principal component.

2.5 Deflation: a theoretical study

A matrix deflation modifies a matrix to eliminate the influence of a given eigenvector, typically by setting the associated eigenvalue to zero. It is a well known technique in linear algebra to find the remaining eigenvalues and eigenvectors. In Mackey (2009) several deflation methods are discussed. All these methods are exact for the eigen-decomposition computation. But when they are used for sparse PCA, due to the fact that typically we do not have true eigenvectors, their performance can be quite different; see Mackey (2009) for an rigorous empirical study. In this section we consider three different deflation methods: Hotelling’s deflation, projection deflation, and Schur complement deflation. We will compare them from a theoretical perspective.

Given $\Sigma_1 = \Sigma$ and the pseudo-eigenvector $\mathbf{u}_1$ that is obtained from a sparse PCA algorithm, the deflation technique allows us to find the next sparse pseudo-eigenvector $\mathbf{u}_2$ as the sparse principal component of some matrix Σ_2, and so on. A simple and popular technique is the Hotelling’s deflation:

$$\Sigma_{t+1}^H = \Sigma_t - \sigma_t \mathbf{u}_t \mathbf{u}_t^\top,$$

where $\sigma_t = \mathbf{u}_t^\top \Sigma_t \mathbf{u}_t$. Since $\mathbf{u}_t$ is not a true eigenvector, Σ_t^H is not necessarily positive semi-definite; see Mackey (2009) for an example. Empirical study shows that its performance is almost uniformly worse than the other two methods.

The project deflation is motivated as follows: assume Σ_t is the covariance matrix of the random variable Y. Given the sparse principal component $\mathbf{u}_t$, consider the covariance matrix of the random variable $(I - \mathbf{u}_t \mathbf{u}_t^\top)Y$, the projection of Y onto the orthocomplement of the subspace spanned by $\mathbf{u}_t$. This results in the matrix

$$\Sigma_{t+1}^P = (I - \mathbf{u}_t \mathbf{u}_t^\top)\Sigma_t (I - \mathbf{u}_t \mathbf{u}_t^\top).$$

The Schur complement deflation is

$$\Sigma_{t+1}^S = \Sigma_t - \Sigma_t \mathbf{u}_t \mathbf{u}_t^\top \Sigma_t / \sigma_t,$$

which is motivated by considering the conditional covariance $var(Y|\mathbf{u}_t^\top Y)$. Both Σ_t^P and Σ_t^S are guaranteed to be positive semi-definite.

For these three deflation matrices, we have the following conclusion:

Proposition 2 *Given any vector* $\mathbf{u}_t$ *we have*

$$\lambda_1(\Sigma_{t+1}^S) \leq \lambda_1(\Sigma_{t+1}^P) \leq \lambda_1(\Sigma_{t+1}^H).$$

However, this result does not mean that the Hotelling's deflation is the best. Instead, empirical study shows it is the worst. We will show that it overestimates the additional variance explained when used for sparse PCA.

Note that one should be careful when dealing with the cumulative variance of multiple sparse PCs. It refers to the variance captured by the subspace spanned by these pseudo-eigenvectors and can be calculated as follows: Denote by P_i the projection onto the subspace spanned by $\mathbf{u}_j,\ j < i$. Let $\mathbf{w}_1 = \mathbf{u}_1$ and sequentially compute

$$\mathbf{w}_{i+1} = \frac{\mathbf{u}_{i+1} - P_i \mathbf{u}_{i+1}}{\|\mathbf{u}_{i+1} - P_i \mathbf{u}_{i+1}\|}.$$

Then $\{\mathbf{w}_j : j \leq i\}$ form an orthonormal basis of subspace spanned by $\{\mathbf{u}_j : j \leq i\}$. The cumulative variance explained by $\mathbf{u}_1, \ldots, \mathbf{u}_t$ is then

$$CV(\mathbf{u}_1, \ldots, \mathbf{u}_t) = \sum_{j=1}^{t} \mathbf{w}_j^\top \Sigma \mathbf{w}_j.$$

The additional variance explained by $\mathbf{u}_{t+1}$ is then

$$\Delta_{t+1} = CV(\mathbf{u}_1, \ldots, \mathbf{u}_t, \mathbf{u}_{t+1}) - CV(\mathbf{u}_1, \ldots, \mathbf{u}_t) = \mathbf{w}_{t+1}^\top \Sigma \mathbf{w}_{t+1}.$$

When the three different deflation are used, we have the following comparison between the additional variance explained.

Proposition 3 *Given unit vectors* $\mathbf{u}_i,\ i \leq t$, *assume that* $\mathbf{u}_{t+1}^H$, $\mathbf{u}_{t+1}^P$ *and* $\mathbf{u}_{t+1}^S$ *are the leading eigenvectors of the three deflated matrices respectively. Then we have*

$$\Delta_{t+1}^P \geq \Delta_{t+1}^H \quad \textit{and} \quad \Delta_{t+1}^P \geq \Delta_{t+1}^S.$$

Although this result is only for the true leading eigenvector, not the sparse pseudo-eigenvector we are looking for, it nevertheless intuitively explains the superiority of the projection deflation method in capturing additional variance, an observation that has been made consistently in practice.

While in practical applications the projection deflation method on average outperforms the other methods, exceptions do occur on rare occasions. We suspect that there are two possible explanations: (1) We do not know whether the superiority of the projection deflation method is still true for sparse PCs. Furthermore, all existing sparse PCA methods solve relaxed problems. The sparse eigenvector is not exact. (2) In Proposition 3 the conclusion is true only when the first t spare eigenvectors are the same. When different deflations are used, only the first pseudo-eigenvector $\mathbf{u}_1$ is the guaranteed to be the same, and it does not guarantee the projection deflation will continue to be better thereafter. The empirical study in Mackey (2009) shows that both the projection deflation and Schur complement deflation outperform the Hotelling's deflation consistently, and the projection deflation is better than or at least comparable with Schur complement deflation in general, especially for the first two or three sparse PCs. This coincides with our theoretical analysis. Therefore, in this study we shall use the projection deflation as the benchmark in our simulations.

3. Simulations

In this section we illustrate the effectiveness of the iterative elimination algorithm on several synthetic and real data sets. We will compare our method with various state-of-the-art methods for sparse PCA.

3.1 Comparison of the criteria

We first compare the AMVL and MAV criteria in the iterative elimination algorithm. We generate a random matrix $M \in \mathbb{R}^{10\times 20}$ with each entry independently drawn from univariate normal distribution with variance 1. The covariance matrix of dimension 10 is then obtained as $\Sigma = MM^{\top}$. We compare the results for the computation of the first sparse PC.

We first compare the performance on one variable deletion, that is, we consider the sparse PC with sparsity $k = 9$. We repeat the experiments 10000 times. In Figure 1 (a) we plot the differences between variances explained by the first sparse PCs obtained from AMVL criterion and the MAV criterion. It shows that in most repeats (94.5%) two criteria yield identical results. In about 500 runs (5%) AMVL criterion outperforms MAV while in about 50 runs (0.5%) MAV outperforms AMVL. Moreover, in the cases that favor AMVL the differences in explained variance can be substantial while in the cases that favor MAV the differences tend to be very small.

Next we compare the performance on sparser PCs. Under the same setting as above we consider the sparse PC with sparsity $k = 5$ instead. Again we run the experiment 10000 times and plot the differences of the variances captured in Figure 1 (b). In about 82% of all repeats the results are the same. This percentage is much lower than in previous setting of $k = 9$. In about 15% of the repeats AMVL has outperformed MAV while in only about 3% of the repeats MAV has performed better. This shows the effect of the elimination criteria cumulates after many iterations.

The above comparisons suggest that on average the iterative elimination algorithm performs better with the AMVL criterion than with the MAV criterion. Because exceptions do occur, in practice a greedy search is suggested in case that two criteria select different variables to eliminate. Notice that this happens with only a very small probability. The additional computation time will not be too much.

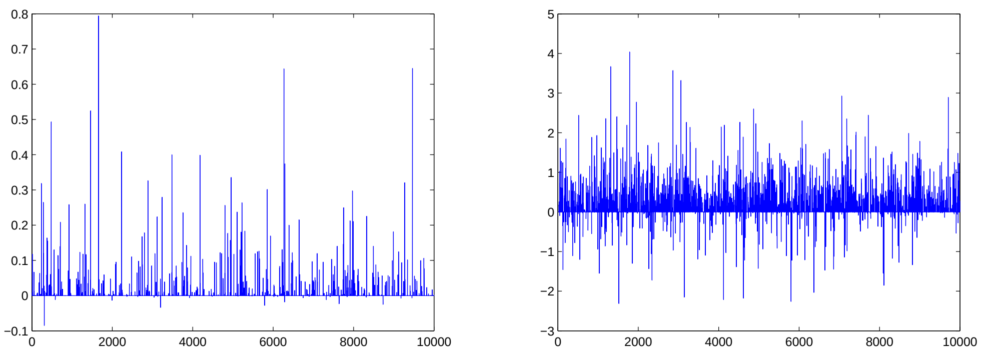


Figure 1: Differences of variances captured by sparse PCs obtained via the AMVL criterion and the simple thresholding MAV criterion. On the left the sparsity is set at $k = 9$. On the right the sparsity is set at $k = 5$.

3.2 A simulated example

The following simulated example was introduced in Zou et al. (2006) and has been used to test the correctness of sparse PCA methods. Three hidden vectors are given as

$$V_1 \sim N(0, 290),\ V_2 \sim N(0, 300),\ V_3 = -0.3V_1 + 0.925V_2 + \epsilon$$

where V_1, V_2 and ϵ are independent and $\epsilon \sim N(0, 1)$. Then 10 observable variables are generated:

$$\begin{aligned} X_i &= V_1 + \epsilon_i, &&\text{for } i = 1, 2, 3, 4 \\ X_i &= V_2 + \epsilon_i, &&\text{for } i = 5, 6, 7, 8 \\ X_i &= V_3 + \epsilon_i &&\text{for } i = 9, 10 \end{aligned}$$

where $\epsilon_i \sim N(0, 1)$ for all i and are independent. The top two sparse PCs with sparsity $k = 4$ are

$$\begin{aligned} \mathbf{u}_1 &= (0, 0, 0, 0, 0.5, 0.5, 0.5, 0.5, 0, 0)^\top, \\ \mathbf{u}_2 &= (0.5, 0.5, 0.5, 0.5, 0, 0, 0, 0, 0, 0)^\top. \end{aligned}$$

Simple thresholding method cannot select the correct variables for the first sparse PC while all the other state-of-the-art methods can (d'Aspremont et al., 2007). With the iterative elimination algorithm, correct result can be attained under both the AMVL and the MAV criteria.

3.3 Pit Props data

The Pit Props data (Jeffers, 1967) has become a benchmark data set for testing the performance of sparse PCA methods. It has 180 observations and 13 variables. The first 6 PCs explain 87%

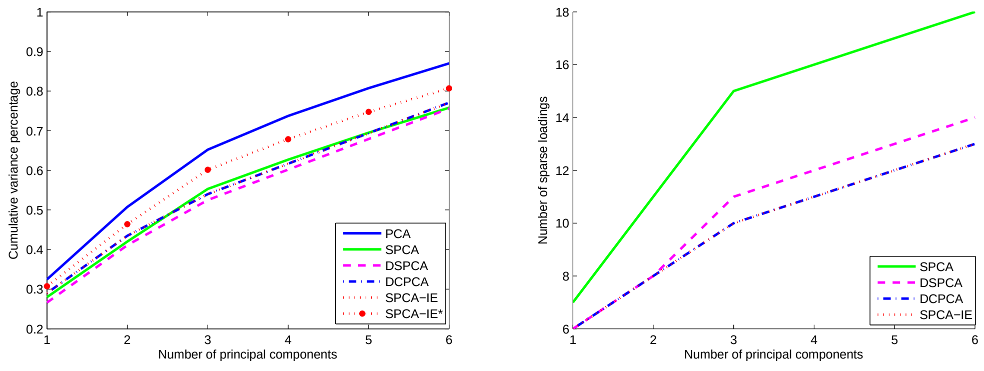


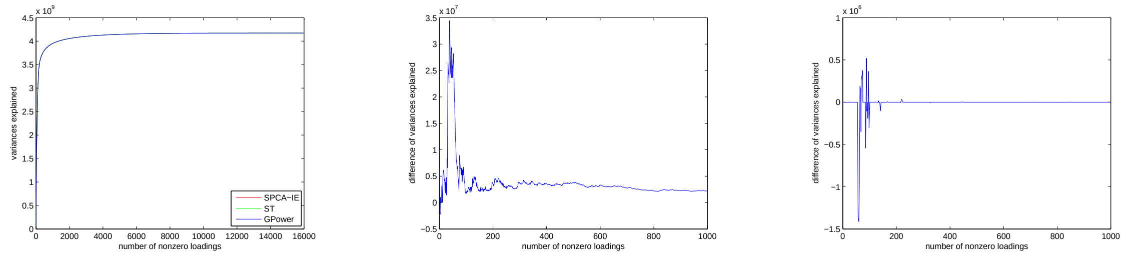
Figure 2: Pit props: (a) cumulative variance and (b) cumulative nonzero loadings of the first 6 sparse principal components. SPCA-IE* is the result for iterative elimination with sparsity $(7, 4, 4, 1, 1, 1)$.

of the total variance. So in the literature the effectiveness of the sparse PCA methods are usually compared using the explanatory power of 6 PCs. SPCA explains 75.8% of the total variance using 6 PCs with sparsity $(7, 4, 4, 1, 1, 1)$ respectively. DSPCA explains 75.5% of the total variance with sparsity pattern $(6, 2, 3, 1, 1, 1)$. GPower explains 76.6% of the total variance with sparsity pattern $(6, 2, 2, 1, 1, 1)$ if ℓ_1 penalty is used and explains 77.2% of the total variance with sparsity pattern $(6, 3, 2, 1, 1, 1)$ if ℓ_0 penalty is used. DC-PCA explains 77.1% of the total variance with sparse pattern $(6, 2, 2, 1, 1, 1)$. We apply our iterative elimination algorithm with sparsity $(6, 2, 2, 1, 1, 1)$, the sparsest setting among the above ones. The 6 sparse PCs explains 77.1% of the total variance, the same as DC-PCA and better than SPCA, DSPCA and GPower with ℓ_1 penalty. It is slightly worse than GPower with ℓ_0 penalty but the latter requires one more nonzero loading. If we apply the sparsity $(7, 4, 4, 1, 1, 1)$ the percentage of the explained variance can be as high as 80.7%. The comparison of the cumulative variance and number of sparse loadings are summarized in Figure 2.

Although DC-PCA and iterative elimination show the same performance, it is interesting to note that they yield different sparse eigenvectors; see Table 1 below.

method	PC	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}
	1	0.444	0.453	0	0	0	0	0.378	0.342	0.403	0.418	0	0	0
SPCA-IE	2	0	0	0.707	0.707	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0.707	0.707	0	0	0	0	0	0	0
	1	0.449	0.459	0	0	0	0	0.374	0.332	0.403	0.419	0	0	0
DC-PCA	2	0	0	0.707	0.707	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0.816	0.578	0	0	0	0	0	0

Table 1: Pit Props: Loadings for the first three sparse PCs.



(a) (b) (c)

Figure 3: Ramaswamy data: (a) Variances explained by the first Sparse PC. (b) The difference of variances explained by the the first sparse PC obtained using iterative elimination and simple thresholding. (b) The difference of variances explained by the the first sparse PC obtained using iterative elimination and GPower. Here GPower refers to GPower with ℓ_0 penalty.

3.4 Gene expression data

The iterative elimination algorithm is viable for "large p small n problem", which is typical for gene expression data. Here we test it on the Ramaswamy data.

Ramaswamy data (Ramaswamy et al., 2001) has 16063 genes and 144 samples. The first principal component captures 45.9% of the total variance. We apply the iterative elimination algorithm and compute the first sparse principal component for different sparsity. The results are compared with simple thresholding method and GPower with ℓ_0 penalty. From Figure 3 (a) we see that three methods perform quite similarly. The differences between the variances explained are less than 0.01% of the total variance. For more careful comparison, we investigate the small differences. In Figure 3 (2) we plot the differences between the variances explained by iterative elimination and simple thresholding when the cardinality of the sparse PC is less than 1000. It shows the iterative elimination is almost consistently better. In Figure 3 (c) we plot the the difference between the variances explained by iterative thresholding and GPower. We see that in most time two methods are the same while in other cases no one can consistently win.

Recall that SPCA performs even worse than simple thresholding (Zou et al., 2006). It requires 2.5% of genes to capture similar variance (40% of the total) as the first true PC. By iterative elimination, we only need about 1.8% of genes to capture 40% of total variance.

4. Conclusions and discussions

In this paper we introduced an iterative elimination method for sparse PCA. It recursively eliminate variables based on certain criterion, which can be either MAV criterion or AMVL criterion, and recalculate the PC on the reduced space. We demonstrate that the new AMVL criterion for the elimination process is in general superior to the MAV criterion, especially when the variance of the first PC is less dominant. Simulations illustrate its effectiveness in both obtaining sparse loadings and explaining variances.

Unlike other state-of-the-art methods, the iterative elimination algorithm does not require advanced optimization processes while achieving comparable performance. We believe it deserves attention due to its simplicity, effectiveness, and viability for very large dimensional data.

Appendix: Proof of propositions

Proof of Proposition 1. Denote by $\mathbf{a} \in \mathbb{R}^p$ the vector with the i-th component $a_i = v_{1,i}$ and all the other components zero. Let $\mathbf{b} = \mathbf{v}_1 - \mathbf{a}$. Then $\|\mathbf{b}\| = 1 - v_{1,i}^2$. Since the i-th component of $\mathbf{b}$ is zero, we have

$$\mathbf{b}^\top \Sigma \mathbf{b} \le \lambda_1(\Sigma^{\setminus i})\|\mathbf{b}\| = (1 - v_{1,i}^2)\lambda_1(\Sigma^{\setminus i}).$$

Using the facts $\mathbf{v}_1 = \mathbf{a} + \mathbf{b}$ and $\Sigma \mathbf{v}_1 = \lambda_1(\Sigma)\mathbf{v}_1$, simple computation gives

$$\begin{aligned}\lambda_1(\Sigma) = \mathbf{v}_1^\top \Sigma \mathbf{v}_1 \quad &= \mathbf{a}^\top \Sigma \mathbf{a} + 2\mathbf{a}^\top \Sigma \mathbf{b} + \mathbf{b}^\top \Sigma \mathbf{b} \\ &= \mathbf{a}^\top \Sigma \mathbf{a} + 2\mathbf{a}^\top \Sigma (\mathbf{v}_1 - \mathbf{a}) + \mathbf{b}^\top \Sigma \mathbf{b} \\ &= 2\lambda_1(\Sigma)\mathbf{a}^\top \mathbf{v}_1 - \mathbf{a}^\top \Sigma \mathbf{a} + \mathbf{b}^\top \Sigma \mathbf{b} \\ &\le 2\lambda_1 v_{1,i}^2 - \Sigma_{ii} v_{1,i}^2 + \lambda_1(\Sigma^{\setminus i})(1 - v_{1,i}^2).\end{aligned}$$

Our conclusion follows by the rearrangement of terms in the above inequality. ■

Proof of Proposition 2 Without loss of generality we only prove for $t = 1$. The proof for $t > 1$ follows the same idea.

For any unit vector $\mathbf{v}$, it can be written as $\mathbf{v} = \alpha \mathbf{u}_1 + \beta \mathbf{w}$ for some unit vector $\mathbf{w}$ that is orthogonal to $\mathbf{u}_1$ and $\alpha^2 + \beta^2 = 1$. Direct computation gives

$$\mathbf{v}^\top \Sigma_2^H \mathbf{v} = \beta^2 \mathbf{w}^\top \Sigma_1 \mathbf{w} + 2\alpha\beta \mathbf{u}_1^\top \Sigma_1 \mathbf{w} \tag{5}$$

$$\mathbf{v}^\top \Sigma_2^P \mathbf{v} = \beta^2 \mathbf{w}^\top \Sigma_1 \mathbf{w} \tag{6}$$

$$\mathbf{v}^\top \Sigma_2^S \mathbf{v} = \beta^2 \mathbf{w}^\top \Sigma_1 \mathbf{w} + \beta^2 (\mathbf{u}_1^\top \Sigma_1 \mathbf{w})^2 / \sigma_1 \tag{7}$$

The conclusion is an easy consequence of the following obvious inequality:

$$\mathbf{v}^\top \Sigma_2^S \mathbf{v} \le \mathbf{v}^\top \Sigma_2^P \le \max(\mathbf{v}\mathbf{v}^\top \Sigma_2^H \mathbf{v}, \tilde{\mathbf{v}}^\top \Sigma_2^H \tilde{\mathbf{v}})$$

where $\tilde{\mathbf{v}} = \alpha \mathbf{u}_1 - \beta \mathbf{w}$ is also a unit vector. ■

Proof of Proposition 3 Again we only consider $t = 1$. Write $\mathbf{u}_2^* = \alpha_* \mathbf{u}_1 + \beta_* \mathbf{w}_*$ where $*$ represent $H,\ P$ or S for three different deflations. Then it is easy to check that $\Delta_2^* = \mathbf{w}_*^\top \Sigma \mathbf{w}_*$. By (6), in order for $\mathbf{u}_2^P$ to be the leading eigenvector of Σ_2^P we should have $\beta_H = 1$. So we must have

$$\Delta_2^H = \mathbf{w}_H^\top \Sigma \mathbf{w}_H = \mathbf{w}_H^\top \Sigma_2^P \mathbf{w}_H \leq (\mathbf{u}_2^P)^\top \Sigma_2^P \mathbf{u}_2^P = \mathbf{w}_P^\top \Sigma \mathbf{w}_P = \Delta_2^P.$$

Similarly we can prove $\Delta_2^S \leq \Delta_2^P$. ■

References

Jorge Cadima and Ian T. Jolliffe. Loadings and correlations in the interpretation of principal components. *Journal of Applied Statistics*, 22(2):203–215, 1995.

Alexandre d'Aspremont, Laurent El Ghaoui, Michael I. Jordan, and Gert R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Rev.*, 49(3):434–448 (electronic), 2007.

Alexandre d'Aspremont, Francis Bach, and Laurent El Ghaoui. Optimal solutions for sparse principal component analysis. *J. Mach. Learn. Res.*, 9:1269–1294, 2008.

G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1983.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

J. Jeffers. Two case studies in the application of principal component. *Applied Statistics*, 16:225–236, 1967.

Ian T. Jolliffe, Nickolay T. Trendafilov, and Mudassir Uddin. A modified principal component technique based on the LASSO. *J. Comput. Graph. Statist.*, 12(3):531–547, 2003.

I.T. Jolliffe. *Principal component analysis*. Springer verlag, 2002.

M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre. Generalized power method for sparse principal component analysis. *Submitted to Journal of Machine Learning Research (preprint available on ArXiv)*, 2008.

Lester Mackey. Deflation methods for sparse PCA. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1017–1024. 2009.

Baback Moghaddam, Yair Weiss, and Shai Avidan. Spectral bounds for sparse PCA: Exact and greedy algorithms. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 915–922. MIT Press, Cambridge, MA, 2006.

S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukheriee, C. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. Mesirov, T. Poggio, W. Gerald, M. Loda, E. Lander, and T. Golub. Multiclass cancer diagnosis using tumor gene expression signature. *Proceedings of the National Academy of Sciences*, 98:15149–15154, 2001.

Haipeng Shen and Jianhua Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *J. Multivariate Anal.*, 99(6):1015–1034, 2008.

B. Sriperumbudur, D. Torres, and G. Lanckriet. A D.C. programming approach to the sparse generalized eigenvalue problem. 2009.

B.K. Sriperumbudur, D.A. Torres, and G.R.G. Lanckriet. Sparse eigen methods by dc programming. In *Proceedings of the 24th International Conference on Machine learning*, pages 831–838, 2007.

R. Zass and A. Shashua. Nonnegative sparse PCA. *Advances in Neural Information Processing Systems*, 19:1561, 2007.

Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *J. Comput. Graph. Statist.*, 15(2):265–286, 2006.